Fashion MNIST CNN Model Analysis

Summary

This project implements a Convolutional Neural Network (CNN) in PyTorch to classify images in the Fashion MNIST dataset, a collection of grayscale images representing various clothing items. By utilizing configurable hyperparameters and training multiple models, the project evaluates the effects of different hyperparameter combinations on model accuracy. Additionally, visualizations like heatmaps and line plots showcase the relationship between hyperparameters and accuracy.

Objective

The primary objective of this project is to:

1. Build a CNN capable of classifying images from the Fashion MNIST dataset with high accuracy.
2. Investigate the impact of varying key hyperparameters, including batch size and number of epochs, on the model's performance.
3. Present results using visualizations to identify optimal hyperparameter settings for future applications.

Process

The project workflow involves the following steps:

1. Data Preparation

The Fashion MNIST dataset was loaded and preprocessed using PyTorch utilities. Each image was converted into a tensor and normalized. The dataset was divided into training and testing sets, with data loaders configured for batch processing. Below is the dataset loading function:

```
train_data = datasets.FashionMNIST(
    root='data',                # Directory to store/download the dataset
    train=True,                 # Flag to indicate this is the training set
    transform=ToTensor(),       # Transform the image data into PyTorch tensors
    download=True,              # Download the dataset if it's not already available
)

test_data = datasets.FashionMNIST(
    root='data',                # Same root directory as the training set
    train=False,                # Flag to indicate this is the testing set
    transform=ToTensor(),       # Apply the same tensor transformation
)

loaders = {
    'train': DataLoader(train_data,batch_size=self.batch_size,shuffle=True,num_workers=self.num_workers,),
    'test': DataLoader(test_data, batch_size=self.batch_size, shuffle=False, num_workers=self.num_workers,),
}
```

2. Model Architecture

A CNN was implemented with two convolutional layers, ReLU activations, max-pooling, and a fully connected output layer for classification. Key hyperparameters (e.g., kernel size, stride, padding) were made configurable via command-line arguments.

3. Training and Evaluation

The models were trained using the Adam optimizer and cross-entropy loss function. Training was performed for varying combinations of epochs and batch sizes. Accuracy was computed on the test dataset after each training session.

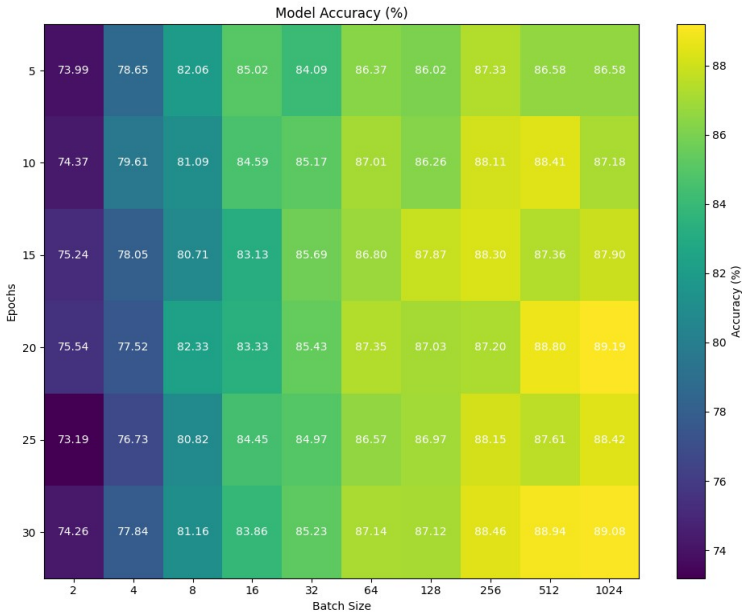
4. Visualization

Heatmaps and line plots were generated to visualize the relationship between hyperparameters and accuracy.

Results

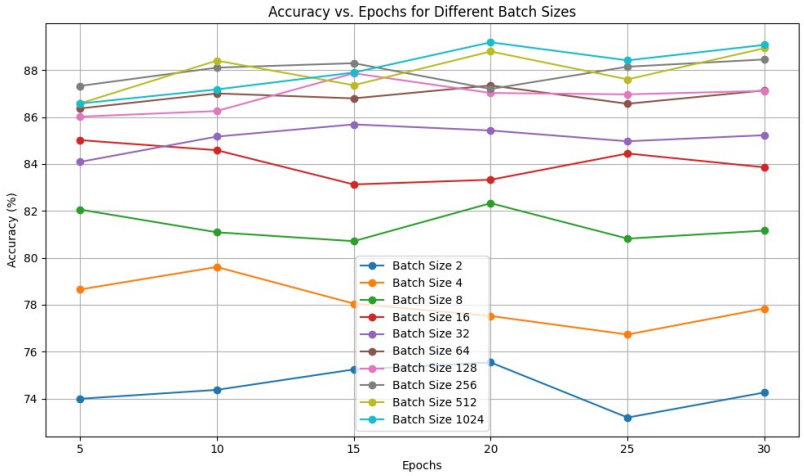
1. Heatmap of Accuracy

A heatmap was generated to display the accuracy of models trained with varying batch sizes and epochs. The figure below shows how accuracy changes with these hyperparameters:



2. Accuracy Trends

A line plot was created to depict accuracy as a function of epochs for different batch sizes. This visualization highlights the performance improvements with increased training epochs:



Observations:

- Higher batch sizes tend to stabilize the training process, but very large batch sizes show diminishing returns.
- Increasing the number of epochs generally improves accuracy, though gains taper off after a certain point.

Conclusion

The project successfully demonstrated the ability of CNNs to classify the Fashion MNIST dataset with high accuracy. It also illustrated how hyperparameters like batch size and number of epochs influence performance. The visualizations provided clear insights into optimal training configurations.

Key Takeaways:

- Smaller batch sizes allow the model to update weights more frequently, but they may lead to noisier gradients.
- Increasing epochs improves accuracy but also increases training time, highlighting the importance of balancing these factors.

Further Steps

To expand on this project, the following steps can be taken:

1. Explore additional hyperparameters like learning rate, dropout, and weight initialization methods.
2. Implement more complex CNN architectures to improve accuracy further.
3. Evaluate performance on other datasets or under real-world conditions.
4. Experiment with data augmentation techniques to improve generalization.

References

- PyTorch Documentation
- Fashion MNIST Dataset
- MNIST with torchvision and skorch
- Python
- Matplotlib: Visualization with Python